Decision Trees: From Intuition to Implementation

Nipun Batra and teaching staff

IIT Gandhinagar

July 30, 2025

Outline

What Are Decision Trees?

- **Intuitive approach:** Make decisions by asking yes/no questions

What Are Decision Trees?

- **Intuitive approach:** Make decisions by asking yes/no questions
- **Tree structure:** Each internal node = decision rule, leaves = predictions

What Are Decision Trees?

- **Intuitive approach:** Make decisions by asking yes/no questions
- **Tree structure:** Each internal node = decision rule, leaves = predictions
- **Human-interpretable:** Easy to understand and explain

What Are Decision Trees?

- **Intuitive approach:** Make decisions by asking yes/no questions
- **Tree structure:** Each internal node = decision rule, leaves = predictions
- **Human-interpretable:** Easy to understand and explain

What Are Decision Trees?

- **Intuitive approach:** Make decisions by asking yes/no questions
- **Tree structure:** Each internal node = decision rule, leaves = predictions
- **Human-interpretable:** Easy to understand and explain
- **Versatile:** Works for both classification and regression

What Are Decision Trees?

- **Intuitive approach:** Make decisions by asking yes/no questions
- **Tree structure:** Each internal node = decision rule, leaves = predictions
- **Human-interpretable:** Easy to understand and explain
- **Versatile:** Works for both classification and regression

What Are Decision Trees?

- **Intuitive approach:** Make decisions by asking yes/no questions
- **Tree structure:** Each internal node = decision rule, leaves = predictions
- **Human-interpretable:** Easy to understand and explain
- **Versatile:** Works for both classification and regression

Real-world Example

Should I play tennis today?

What Are Decision Trees?

- **Intuitive approach:** Make decisions by asking yes/no questions
- **Tree structure:** Each internal node = decision rule, leaves = predictions
- **Human-interpretable:** Easy to understand and explain
- **Versatile:** Works for both classification and regression

Real-world Example

Should I play tennis today?

- If sunny $\rightarrow$ check humidity

What Are Decision Trees?

- **Intuitive approach:** Make decisions by asking yes/no questions
- **Tree structure:** Each internal node = decision rule, leaves = predictions
- **Human-interpretable:** Easy to understand and explain
- **Versatile:** Works for both classification and regression

Real-world Example

Should I play tennis today?

- If sunny $\rightarrow$ check humidity
- If rainy $\rightarrow$ check wind

What Are Decision Trees?

- **Intuitive approach:** Make decisions by asking yes/no questions
- **Tree structure:** Each internal node = decision rule, leaves = predictions
- **Human-interpretable:** Easy to understand and explain
- **Versatile:** Works for both classification and regression

Real-world Example

Should I play tennis today?

- If sunny $\rightarrow$ check humidity
- If rainy $\rightarrow$ check wind
- If overcast $\rightarrow$ always play!

Pop Quiz: Decision Tree Basics

Quick Quiz 1

What makes decision trees particularly useful in practice?

a) They always give the highest accuracy

Answer: b) Interpretability is a key advantage of decision trees!

Pop Quiz: Decision Tree Basics

Quick Quiz 1

What makes decision trees particularly useful in practice?

a) They always give the highest accuracy
b) They're easy to interpret and explain to humans

Answer: b) Interpretability is a key advantage of decision trees!

Pop Quiz: Decision Tree Basics

Quick Quiz 1

What makes decision trees particularly useful in practice?

a) They always give the highest accuracy
b) They're easy to interpret and explain to humans
c) They work only for numerical data

Answer: b) Interpretability is a key advantage of decision trees!

Information Gain Intuition

The Core Question: How do we choose the best question to ask?

- Dataset: 9 Yes, 5 No (mixed outcomes)

Information Gain Intuition

The Core Question: How do we choose the best question to ask?

- Dataset: 9 Yes, 5 No (mixed outcomes)

Information Gain Intuition

The Core Question: How do we choose the best question to ask?

- Dataset: 9 Yes, 5 No (mixed outcomes)
- Scenario 1: What if we had 14 Yes, 0 No?

Information Gain Intuition

The Core Question: How do we choose the best question to ask?

- Dataset: 9 Yes, 5 No (mixed outcomes)
- Scenario 1: What if we had 14 Yes, 0 No?

Information Gain Intuition

The Core Question: How do we choose the best question to ask?

- Dataset: 9 Yes, 5 No (mixed outcomes)
- Scenario 1: What if we had 14 Yes, 0 No?
- Scenario 2: What if we had 0 Yes, 14 No?

Information Gain Intuition

The Core Question: How do we choose the best question to ask?

- Dataset: 9 Yes, 5 No (mixed outcomes)
- Scenario 1: What if we had 14 Yes, 0 No?
- Scenario 2: What if we had 0 Yes, 14 No?

Information Gain Intuition

The Core Question: How do we choose the best question to ask?

- Dataset: 9 Yes, 5 No (mixed outcomes)
- Scenario 1: What if we had 14 Yes, 0 No?
- Scenario 2: What if we had 0 Yes, 14 No?
- **Key insight:** Pure subsets (no disagreement) are *easier* to handle!

Information Gain Intuition

The Core Question: How do we choose the best question to ask?

- Dataset: 9 Yes, 5 No (mixed outcomes)
- Scenario 1: What if we had 14 Yes, 0 No?
- Scenario 2: What if we had 0 Yes, 14 No?
- **Key insight:** Pure subsets (no disagreement) are *easier* to handle!

Information Gain Intuition

The Core Question: How do we choose the best question to ask?

- Dataset: 9 Yes, 5 No (mixed outcomes)
- Scenario 1: What if we had 14 Yes, 0 No?
- Scenario 2: What if we had 0 Yes, 14 No?
- **Key insight:** Pure subsets (no disagreement) are *easier* to handle!
- We need a statistical measure of "disagreement" or "impurity"

Information Gain Intuition

The Core Question: How do we choose the best question to ask?

- Dataset: 9 Yes, 5 No (mixed outcomes)
- Scenario 1: What if we had 14 Yes, 0 No?
- Scenario 2: What if we had 0 Yes, 14 No?
- **Key insight:** Pure subsets (no disagreement) are *easier* to handle!
- We need a statistical measure of "disagreement" or "impurity"

Information Gain Intuition

The Core Question: How do we choose the best question to ask?

- Dataset: 9 Yes, 5 No (mixed outcomes)
- Scenario 1: What if we had 14 Yes, 0 No?
- Scenario 2: What if we had 0 Yes, 14 No?
- **Key insight:** Pure subsets (no disagreement) are *easier* to handle!
- We need a statistical measure of "disagreement" or "impurity"

Goal

Find the question that **reduces impurity the most** after splitting

Entropy: Measuring Impurity

Entropy: Measuring Impurity

Interpretation:

Entropy: Measuring Impurity

Entropy Formula

$$H(X) = -\sum_{i=1}^{k} p(x_i) \log_2 p(x_i)$$

Interpretation:

Entropy: Measuring Impurity

Entropy Formula

$$H(X) = -\sum_{i=1}^{k} p(x_i) \log_2 p(x_i)$$

Interpretation:

- $H(X) = 0$: Pure set (all same class)

Entropy: Measuring Impurity

Entropy Formula

$$H(X) = -\sum_{i=1}^{k} p(x_i)\log_2 p(x_i)$$

Interpretation:

- $H(X) = 0$: Pure set (all same class)
- $H(X) = 1$: Maximum impurity (50-50 split)

Entropy: Measuring Impurity

Entropy Formula

$$H(X) = -\sum_{i=1}^{k} p(x_i) \log_2 p(x_i)$$

Interpretation:

- $H(X) = 0$: Pure set (all same class)
- $H(X) = 1$: Maximum impurity (50-50 split)
- Higher entropy = more mixed/impure

Entropy: Measuring Impurity

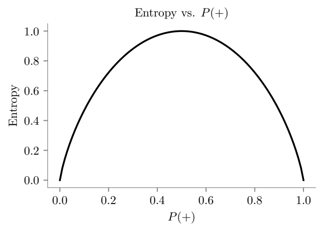
Entropy Formula

$$H(X) = -\sum_{i=1}^{k} p(x_i) \log_2 p(x_i)$$

Interpretation:

- $H(X) = 0$: Pure set (all same class)
- $H(X) = 1$: Maximum impurity (50-50 split)
- Higher entropy = more mixed/impure

Notebook: entropy.html



Entropy: Measuring Impurity

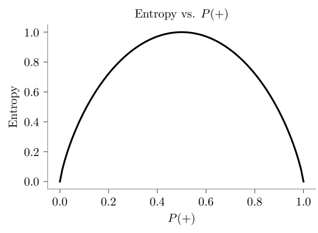
Entropy Formula

$$H(X) = -\sum_{i=1}^{k} p(x_i) \log_2 p(x_i)$$

Interpretation:

- $H(X) = 0$: Pure set (all same class)
- $H(X) = 1$: Maximum impurity (50-50 split)
- Higher entropy = more mixed/impure

Notebook: entropy.html



Entropy: Measuring Impurity

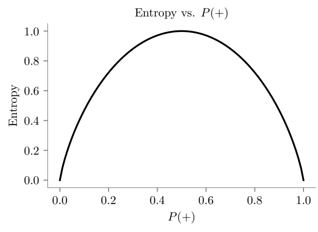
Entropy Formula

$$H(X) = -\sum_{i=1}^{k} p(x_i) \log_2 p(x_i)$$

Interpretation:

- $H(X) = 0$: Pure set (all same class)
- $H(X) = 1$: Maximum impurity (50-50 split)
- Higher entropy = more mixed/impure

Notebook: entropy.html



Pop Quiz: Entropy Calculation

Quick Quiz 2

For a dataset with 8 positive and 8 negative examples, what is the entropy?

a) $H = 0$ (pure)

Answer: b) $H = -0.5\log_2(0.5) - 0.5\log_2(0.5) = 1$

Pop Quiz: Entropy Calculation

Quick Quiz 2

For a dataset with 8 positive and 8 negative examples, what is the entropy?

a) $H = 0$ (pure)
b) $H = 1$ (maximum impurity)

Answer: b) $H = -0.5\log_2(0.5) - 0.5\log_2(0.5) = 1$

Pop Quiz: Entropy Calculation

Quick Quiz 2

For a dataset with 8 positive and 8 negative examples, what is the entropy?

a) $H = 0$ (pure)
b) $H = 1$ (maximum impurity)
c) $H = 0.5$ (moderate impurity)

Answer: b) $H = -0.5\log_2(0.5) - 0.5\log_2(0.5) = 1$

Choosing the Root Node

Question: Which feature should we use as the root node?

- Goal: Choose the feature that best separates the classes

Choosing the Root Node

Question: Which feature should we use as the root node?

- Goal: Choose the feature that best separates the classes

Choosing the Root Node

Question: Which feature should we use as the root node?

- Goal: Choose the feature that best separates the classes
- Example: When Outlook = Overcast, we always Play!

Choosing the Root Node

Question: Which feature should we use as the root node?

- Goal: Choose the feature that best separates the classes
- Example: When Outlook = Overcast, we always Play!
  - Perfect separation = zero disagreement

Choosing the Root Node

Question: Which feature should we use as the root node?

- Goal: Choose the feature that best separates the classes
- Example: When Outlook = Overcast, we always Play!
  - Perfect separation = zero disagreement
  - This is a *very good* feature for splitting

Choosing the Root Node

Question: Which feature should we use as the root node?

- Goal: Choose the feature that best separates the classes
- Example: When Outlook = Overcast, we always Play!
  - Perfect separation = zero disagreement
  - This is a *very good* feature for splitting

Choosing the Root Node

Question: Which feature should we use as the root node?

- Goal: Choose the feature that best separates the classes
- Example: When Outlook = Overcast, we always Play!
  - Perfect separation = zero disagreement
  - This is a *very good* feature for splitting
- Criterion: Select feature with highest **Information Gain**

Choosing the Root Node

Question: Which feature should we use as the root node?

- Goal: Choose the feature that best separates the classes
- Example: When Outlook = Overcast, we always Play!
  - Perfect separation = zero disagreement
  - This is a *very good* feature for splitting
- Criterion: Select feature with highest **Information Gain**

Choosing the Root Node

Question: Which feature should we use as the root node?

- Goal: Choose the feature that best separates the classes
- Example: When Outlook = Overcast, we always Play!
  - Perfect separation = zero disagreement
  - This is a *very good* feature for splitting
- Criterion: Select feature with highest **Information Gain**

Information Gain Formula

$$\text{Gain}(S, A) = H(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v)$$

Entropy Calculation Examples

Entropy Calculation Examples

Entropy Calculation Examples

Pure Subset

Outlook	Play
Overcast	Yes
Overcast	Yes
Overcast	Yes
Overcast	Yes

4 Yes, 0 No

$H = 0$ (perfect purity!)

Entropy Calculation Examples

Pure Subset

Outlook	Play
Overcast	Yes
Overcast	Yes
Overcast	Yes
Overcast	Yes

4 Yes, 0 No

$H = 0$ (perfect purity!)

Entropy Calculation Examples

Pure Subset

Outlook	**Play**
Overcast	Yes
Overcast	Yes
Overcast	Yes
Overcast	Yes

4 Yes, 0 No

$H = 0$ (perfect purity!)

Entropy Calculation Examples

Pure Subset

Outlook	**Play**
Overcast	Yes
Overcast	Yes
Overcast	Yes
Overcast	Yes

4 Yes, 0 No

$H = 0$ (perfect purity!)

Mixed Subset

Outlook	**Play**
Rain	Yes
Rain	Yes
Rain	No
Rain	Yes
Rain	No

3 Yes, 2 No

$H = -\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right)$

$= 0.971$ (high impurity)

Entropy Calculation Examples

Pure Subset

Outlook	Play
Overcast	Yes
Overcast	Yes
Overcast	Yes
Overcast	Yes

4 Yes, 0 No

$H = 0$ (perfect purity!)

Mixed Subset

Outlook	Play
Rain	Yes
Rain	Yes
Rain	No
Rain	Yes
Rain	No

3 Yes, 2 No

$H = -\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right)$
$= 0.971$ (high impurity)

Entropy Calculation Examples

Pure Subset

Outlook	Play
Overcast	Yes
Overcast	Yes
Overcast	Yes
Overcast	Yes

4 Yes, 0 No

$H = 0$ (perfect purity!)

Mixed Subset

Outlook	Play
Rain	Yes
Rain	Yes
Rain	No
Rain	Yes
Rain	No

3 Yes, 2 No

$H = -\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right)$
$= 0.971$ (high impurity)

Key Insight: We want to create splits that result in low entropy subsets!

Information Gain: Step-by-Step

For Outlook=Sunny subset (2 Yes, 3 No):

- Temp split:

Information Gain: Step-by-Step

For Outlook=Sunny subset (2 Yes, 3 No):

- Temp split:
    - Hot: (0 Yes, 2 No), Cool: (1 Yes, 1 No), Mild: (1 Yes, 0 No)

Information Gain: Step-by-Step

For Outlook=Sunny subset (2 Yes, 3 No):

- Temp split:
  - Hot: (0 Yes, 2 No), Cool: (1 Yes, 1 No), Mild: (1 Yes, 0 No)
  - Gain = $H(2,3) - \frac{2}{5}H(0,2) - \frac{2}{5}H(1,1) - \frac{1}{5}H(1,0)$

Information Gain: Step-by-Step

For Outlook=Sunny subset (2 Yes, 3 No):

- Temp split:
  - Hot: (0 Yes, 2 No), Cool: (1 Yes, 1 No), Mild: (1 Yes, 0 No)
  - Gain = $H(2,3) - \frac{2}{5}H(0,2) - \frac{2}{5}H(1,1) - \frac{1}{5}H(1,0)$

Information Gain: Step-by-Step

For Outlook=Sunny subset (2 Yes, 3 No):

- Temp split:
  - Hot: (0 Yes, 2 No), Cool: (1 Yes, 1 No), Mild: (1 Yes, 0 No)
  - Gain = $H(2,3) - \frac{2}{5}H(0,2) - \frac{2}{5}H(1,1) - \frac{1}{5}H(1,0)$
- Humidity split:

Information Gain: Step-by-Step

For Outlook=Sunny subset (2 Yes, 3 No):

- Temp split:
  - Hot: (0 Yes, 2 No), Cool: (1 Yes, 1 No), Mild: (1 Yes, 0 No)
  - Gain = $H(2,3) - \frac{2}{5}H(0,2) - \frac{2}{5}H(1,1) - \frac{1}{5}H(1,0)$
- Humidity split:
  - High: (0 Yes, 3 No), Normal: (2 Yes, 0 No)

Information Gain: Step-by-Step

For Outlook=Sunny subset (2 Yes, 3 No):

- Temp split:
  - Hot: (0 Yes, 2 No), Cool: (1 Yes, 1 No), Mild: (1 Yes, 0 No)
  - Gain = $H(2,3) - \frac{2}{5}H(0,2) - \frac{2}{5}H(1,1) - \frac{1}{5}H(1,0)$
- Humidity split:
  - High: (0 Yes, 3 No), Normal: (2 Yes, 0 No)
  - Gain = $H(2,3) - \frac{3}{5}H(0,3) - \frac{2}{5}H(2,0) =$ **maximum!**

Information Gain: Step-by-Step

For Outlook=Sunny subset (2 Yes, 3 No):

- Temp split:
  - Hot: (0 Yes, 2 No), Cool: (1 Yes, 1 No), Mild: (1 Yes, 0 No)
  - Gain = $H(2,3) - \frac{2}{5}H(0,2) - \frac{2}{5}H(1,1) - \frac{1}{5}H(1,0)$
- Humidity split:
  - High: (0 Yes, 3 No), Normal: (2 Yes, 0 No)
  - Gain = $H(2,3) - \frac{3}{5}H(0,3) - \frac{2}{5}H(2,0) =$ **maximum!**

Information Gain: Step-by-Step

For Outlook=Sunny subset (2 Yes, 3 No):

- Temp split:
  - Hot: (0 Yes, 2 No), Cool: (1 Yes, 1 No), Mild: (1 Yes, 0 No)
  - Gain = $H(2,3) - \frac{2}{5}H(0,2) - \frac{2}{5}H(1,1) - \frac{1}{5}H(1,0)$
- Humidity split:
  - High: (0 Yes, 3 No), Normal: (2 Yes, 0 No)
  - Gain = $H(2,3) - \frac{3}{5}H(0,3) - \frac{2}{5}H(2,0) =$ **maximum!**
- Wind split:

Information Gain: Step-by-Step

For Outlook=Sunny subset (2 Yes, 3 No):

- Temp split:
  - Hot: (0 Yes, 2 No), Cool: (1 Yes, 1 No), Mild: (1 Yes, 0 No)
  - Gain = $H(2,3) - \frac{2}{5}H(0,2) - \frac{2}{5}H(1,1) - \frac{1}{5}H(1,0)$
- Humidity split:
  - High: (0 Yes, 3 No), Normal: (2 Yes, 0 No)
  - Gain = $H(2,3) - \frac{3}{5}H(0,3) - \frac{2}{5}H(2,0) =$ **maximum!**
- Wind split:
  - Strong: (1 Yes, 2 No), Weak: (1 Yes, 1 No)

Information Gain: Step-by-Step

For Outlook=Sunny subset (2 Yes, 3 No):

- Temp split:
  - Hot: (0 Yes, 2 No), Cool: (1 Yes, 1 No), Mild: (1 Yes, 0 No)
  - Gain = $H(2,3) - \frac{2}{5}H(0,2) - \frac{2}{5}H(1,1) - \frac{1}{5}H(1,0)$
- Humidity split:
  - High: (0 Yes, 3 No), Normal: (2 Yes, 0 No)
  - Gain = $H(2,3) - \frac{3}{5}H(0,3) - \frac{2}{5}H(2,0) =$ **maximum!**
- Wind split:
  - Strong: (1 Yes, 2 No), Weak: (1 Yes, 1 No)
  - Lower gain than Humidity

Information Gain: Step-by-Step

For Outlook=Sunny subset (2 Yes, 3 No):

- Temp split:
  - Hot: (0 Yes, 2 No), Cool: (1 Yes, 1 No), Mild: (1 Yes, 0 No)
  - Gain = $H(2,3) - \frac{2}{5}H(0,2) - \frac{2}{5}H(1,1) - \frac{1}{5}H(1,0)$
- Humidity split:
  - High: (0 Yes, 3 No), Normal: (2 Yes, 0 No)
  - Gain = $H(2,3) - \frac{3}{5}H(0,3) - \frac{2}{5}H(2,0) =$ **maximum!**
- Wind split:
  - Strong: (1 Yes, 2 No), Weak: (1 Yes, 1 No)
  - Lower gain than Humidity

Information Gain: Step-by-Step

For Outlook=Sunny subset (2 Yes, 3 No):

- Temp split:
  - Hot: (0 Yes, 2 No), Cool: (1 Yes, 1 No), Mild: (1 Yes, 0 No)
  - Gain = $H(2,3) - \frac{2}{5}H(0,2) - \frac{2}{5}H(1,1) - \frac{1}{5}H(1,0)$
- Humidity split:
  - High: (0 Yes, 3 No), Normal: (2 Yes, 0 No)
  - Gain = $H(2,3) - \frac{3}{5}H(0,3) - \frac{2}{5}H(2,0) =$ **maximum!**
- Wind split:
  - Strong: (1 Yes, 2 No), Weak: (1 Yes, 1 No)
  - Lower gain than Humidity

Winner

Humidity gives the highest information gain for this subset!

Pop Quiz: Information Gain

Quick Quiz 3

Which split would give the highest information gain?

a) Split that creates subsets: (5 Yes, 5 No) and (3 Yes, 2 No)

Answer: b) Pure subsets (entropy = 0) give maximum information gain!

Pop Quiz: Information Gain

Quick Quiz 3

Which split would give the highest information gain?

a) Split that creates subsets: (5 Yes, 5 No) and (3 Yes, 2 No)
b) Split that creates subsets: (8 Yes, 0 No) and (0 Yes, 7 No)

Answer: b) Pure subsets (entropy = 0) give maximum information gain!

Pop Quiz: Information Gain

Quick Quiz 3

Which split would give the highest information gain?

a) Split that creates subsets: (5 Yes, 5 No) and (3 Yes, 2 No)

b) Split that creates subsets: (8 Yes, 0 No) and (0 Yes, 7 No)

c) Split that creates subsets: (4 Yes, 3 No) and (4 Yes, 4 No)

Answer: b) Pure subsets (entropy = 0) give maximum information gain!

Prediction Example

Prediction for $<$High Humidity, Strong Wind, Sunny Outlook, Hot Temp$>$ is ?

Prediction Example

Prediction for $<$High Humidity, Strong Wind, Sunny Outlook, Hot Temp$>$ is ?
No

Depth-Limited Trees

Apply the same rules, except when depth limit is reached, the leaf node is assigned the most common occurring value in that path.

Depth-Limited Trees

Apply the same rules, except when depth limit is reached, the leaf node is assigned the most common occurring value in that path.
What is depth-0 tree (no decision) for the examples?

Depth-Limited Trees

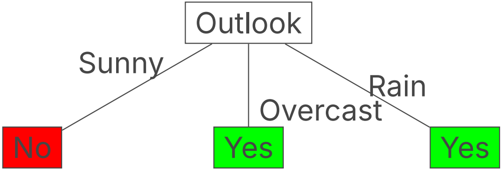
Apply the same rules, except when depth limit is reached, the leaf node is assigned the most common occurring value in that path.
What is depth-0 tree (no decision) for the examples?
Always predicting Yes

Depth-Limited Trees

Apply the same rules, except when depth limit is reached, the leaf node is assigned the most common occurring value in that path.
What is depth-0 tree (no decision) for the examples?
Always predicting Yes
What is depth-1 tree (no decision) for the examples?

Depth-Limited Trees

Apply the same rules, except when depth limit is reached, the leaf node is assigned the most common occurring value in that path.
What is depth-0 tree (no decision) for the examples?
Always predicting Yes
What is depth-1 tree (no decision) for the examples?



Why Outlook is Good Root?

Answer: B) When Outlook=Overcast, all examples have Play=Yes - This creates a pure subset with entropy=0, maximizing information gain.

Regression Trees

- Any guesses?

Regression Trees

- Any guesses?

Regression Trees

- Any guesses?

Regression Trees

- Any guesses?
- Mean Squared Error

Regression Trees

- Any guesses?
- Mean Squared Error

Regression Trees

- Any guesses?
- Mean Squared Error
- $\mathrm{MSE}(S) = 311.34$

Regression Trees

- Any guesses?
- Mean Squared Error
- $\mathrm{MSE}(S) = 311.34$

Regression Trees

- Any guesses?
- Mean Squared Error
- $\mathrm{MSE}(S) = 311.34$
- What about splitting criterion for regression?

Regression Trees

- Any guesses?
- Mean Squared Error
- $\mathrm{MSE}(S) = 311.34$
- What about splitting criterion for regression?

Regression Trees

- Any guesses?
- Mean Squared Error
- $\mathrm{MSE}(S) = 311.34$
- What about splitting criterion for regression?
- **MSE Reduction** (not Information Gain!)

Regression Trees

- Any guesses?
- Mean Squared Error
- $\mathrm{MSE}(S) = 311.34$
- What about splitting criterion for regression?
- **MSE Reduction** (not Information Gain!)

Regression Trees

- Any guesses?
- Mean Squared Error
- $\mathrm{MSE}(S) = 311.34$
- What about splitting criterion for regression?
- **MSE Reduction** (not Information Gain!)
- MSE Reduction = $\mathrm{MSE}(S) - \sum_v \frac{|S_v|}{|S|} \mathrm{MSE}(S_v)$

Regression Splitting Criterion

Answer: C) Mean Squared Error (MSE) Reduction - For regression, we minimize MSE instead of maximizing information gain.

Continuous Features

Answer: B) Use midpoints between consecutive sorted feature values - This ensures we test all meaningful boundaries between different class regions.

Leaf Node Predictions

Answer: C) The mean of target values in that region - Each leaf predicts the average target value of training samples that reach that leaf.

The Problem: Overfitting in Decision Trees

- **Unpruned trees**: Can grow very deep and complex

The Problem: Overfitting in Decision Trees

- **Unpruned trees**: Can grow very deep and complex

The Problem: Overfitting in Decision Trees

- **Unpruned trees**: Can grow very deep and complex
- **Perfect training accuracy**: Each leaf contains single training example

The Problem: Overfitting in Decision Trees

- **Unpruned trees**: Can grow very deep and complex
- **Perfect training accuracy**: Each leaf contains single training example

The Problem: Overfitting in Decision Trees

- **Unpruned trees**: Can grow very deep and complex
- **Perfect training accuracy**: Each leaf contains single training example
- **But**: Poor generalization to new data

The Problem: Overfitting in Decision Trees

- **Unpruned trees**: Can grow very deep and complex
- **Perfect training accuracy**: Each leaf contains single training example
- **But**: Poor generalization to new data

The Problem: Overfitting in Decision Trees

- **Unpruned trees**: Can grow very deep and complex
- **Perfect training accuracy**: Each leaf contains single training example
- **But**: Poor generalization to new data
- **Symptoms**:

The Problem: Overfitting in Decision Trees

- **Unpruned trees**: Can grow very deep and complex
- **Perfect training accuracy**: Each leaf contains single training example
- **But**: Poor generalization to new data
- **Symptoms**:
  - High training accuracy, low test accuracy

The Problem: Overfitting in Decision Trees

- **Unpruned trees**: Can grow very deep and complex
- **Perfect training accuracy**: Each leaf contains single training example
- **But**: Poor generalization to new data
- **Symptoms**:
  - High training accuracy, low test accuracy

The Problem: Overfitting in Decision Trees

- **Unpruned trees**: Can grow very deep and complex
- **Perfect training accuracy**: Each leaf contains single training example
- **But**: Poor generalization to new data
- **Symptoms**:
  - High training accuracy, low test accuracy
  - Very deep trees with many leaves

The Problem: Overfitting in Decision Trees

- **Unpruned trees**: Can grow very deep and complex
- **Perfect training accuracy**: Each leaf contains single training example
- **But**: Poor generalization to new data
- **Symptoms**:
    - High training accuracy, low test accuracy
    - Very deep trees with many leaves
    - Rules that are too specific to training data

The Problem: Overfitting in Decision Trees

- **Unpruned trees**: Can grow very deep and complex
- **Perfect training accuracy**: Each leaf contains single training example
- **But**: Poor generalization to new data
- **Symptoms**:
  - High training accuracy, low test accuracy
  - Very deep trees with many leaves
  - Rules that are too specific to training data
- **Solution**: Pruning to control model complexity

Pre-pruning (Early Stopping)

Stop growing tree before it becomes too complex:

- **Maximum depth**: Limit tree depth (e.g., max_depth = 5)

Pre-pruning (Early Stopping)

Stop growing tree before it becomes too complex:

- **Maximum depth**: Limit tree depth (e.g., max_depth = 5)

Pre-pruning (Early Stopping)

Stop growing tree before it becomes too complex:

- **Maximum depth**: Limit tree depth (e.g., max_depth = 5)
- **Minimum samples per split**: Don't split if node has < N samples

Pre-pruning (Early Stopping)

Stop growing tree before it becomes too complex:

- **Maximum depth**: Limit tree depth (e.g., max_depth = 5)
- **Minimum samples per split**: Don't split if node has < N samples

Pre-pruning (Early Stopping)

Stop growing tree before it becomes too complex:

- **Maximum depth**: Limit tree depth (e.g., max_depth = 5)
- **Minimum samples per split**: Don't split if node has < N samples
- **Minimum samples per leaf**: Ensure each leaf has $\geq$ M samples

Pre-pruning (Early Stopping)

Stop growing tree before it becomes too complex:

- **Maximum depth**: Limit tree depth (e.g., max_depth = 5)
- **Minimum samples per split**: Don't split if node has < N samples
- **Minimum samples per leaf**: Ensure each leaf has $\geq$ M samples

Pre-pruning (Early Stopping)

Stop growing tree before it becomes too complex:

- **Maximum depth**: Limit tree depth (e.g., max_depth = 5)
- **Minimum samples per split**: Don't split if node has < N samples
- **Minimum samples per leaf**: Ensure each leaf has $\geq$ M samples
- **Maximum features**: Consider only subset of features at each split

Pre-pruning (Early Stopping)

Stop growing tree before it becomes too complex:

- **Maximum depth**: Limit tree depth (e.g., max_depth = 5)
- **Minimum samples per split**: Don't split if node has < N samples
- **Minimum samples per leaf**: Ensure each leaf has $\geq$ M samples
- **Maximum features**: Consider only subset of features at each split

Pre-pruning (Early Stopping)

Stop growing tree before it becomes too complex:

- **Maximum depth**: Limit tree depth (e.g., max_depth = 5)
- **Minimum samples per split**: Don't split if node has < N samples
- **Minimum samples per leaf**: Ensure each leaf has $\geq$ M samples
- **Maximum features**: Consider only subset of features at each split
- **Minimum impurity decrease**: Only split if improvement > threshold

Advantages: Simple, computationally efficient
Disadvantages: May stop too early, miss good splits later

Post-pruning (Tree Simplification)

Grow full tree, then remove unnecessary branches:

- **Algorithm**:

Post-pruning (Tree Simplification)

Grow full tree, then remove unnecessary branches:

- **Algorithm**:
  1. Grow complete tree on training data

Post-pruning (Tree Simplification)

Grow full tree, then remove unnecessary branches:

- **Algorithm**:
  1. Grow complete tree on training data

Post-pruning (Tree Simplification)

Grow full tree, then remove unnecessary branches:

- **Algorithm**:
  1. Grow complete tree on training data
  2. Use validation set to evaluate subtree performance

Post-pruning (Tree Simplification)

Grow full tree, then remove unnecessary branches:

- **Algorithm**:
  1. Grow complete tree on training data
  2. Use validation set to evaluate subtree performance
  3. Remove branches that don't improve validation accuracy

Post-pruning (Tree Simplification)

Grow full tree, then remove unnecessary branches:

- **Algorithm**:
  1. Grow complete tree on training data
  2. Use validation set to evaluate subtree performance
  3. Remove branches that don't improve validation accuracy
  4. Repeat until no beneficial removals remain

Post-pruning (Tree Simplification)

Grow full tree, then remove unnecessary branches:

- **Algorithm**:
  1. Grow complete tree on training data
  2. Use validation set to evaluate subtree performance
  3. Remove branches that don't improve validation accuracy
  4. Repeat until no beneficial removals remain

Post-pruning (Tree Simplification)

Grow full tree, then remove unnecessary branches:

- **Algorithm**:
  1. Grow complete tree on training data
  2. Use validation set to evaluate subtree performance
  3. Remove branches that don't improve validation accuracy
  4. Repeat until no beneficial removals remain
- **Cost Complexity Pruning**: Minimize Error $+ \alpha \times$ Tree Size

Post-pruning (Tree Simplification)

Grow full tree, then remove unnecessary branches:

- **Algorithm**:
  1. Grow complete tree on training data
  2. Use validation set to evaluate subtree performance
  3. Remove branches that don't improve validation accuracy
  4. Repeat until no beneficial removals remain
- **Cost Complexity Pruning**: Minimize $\text{Error} + \alpha \times \text{Tree Size}$

Post-pruning (Tree Simplification)

Grow full tree, then remove unnecessary branches:

- **Algorithm**:
  1. Grow complete tree on training data
  2. Use validation set to evaluate subtree performance
  3. Remove branches that don't improve validation accuracy
  4. Repeat until no beneficial removals remain
- **Cost Complexity Pruning**: Minimize Error $+ \alpha \times$ Tree Size
- **Advantages**: More thorough, can recover from early stopping mistakes

Post-pruning (Tree Simplification)

Grow full tree, then remove unnecessary branches:

- **Algorithm**:
  1. Grow complete tree on training data
  2. Use validation set to evaluate subtree performance
  3. Remove branches that don't improve validation accuracy
  4. Repeat until no beneficial removals remain
- **Cost Complexity Pruning**: Minimize $\text{Error} + \alpha \times \text{Tree Size}$
- **Advantages**: More thorough, can recover from early stopping mistakes

Post-pruning (Tree Simplification)

Grow full tree, then remove unnecessary branches:

- **Algorithm**:
  1. Grow complete tree on training data
  2. Use validation set to evaluate subtree performance
  3. Remove branches that don't improve validation accuracy
  4. Repeat until no beneficial removals remain
- **Cost Complexity Pruning**: Minimize $\text{Error} + \alpha \times \text{Tree Size}$
- **Advantages**: More thorough, can recover from early stopping mistakes
- **Disadvantages**: More computationally expensive

Cost Complexity Pruning Algorithm

Systematic approach to find optimal tree size:

- **Cost function:** $R_\alpha(T) = R(T) + \alpha|T|$

Cost Complexity Pruning Algorithm

Systematic approach to find optimal tree size:

- **Cost function**: $R_\alpha(T) = R(T) + \alpha|T|$
  - $R(T)$: Misclassification error on validation set

Cost Complexity Pruning Algorithm

Systematic approach to find optimal tree size:

- **Cost function**: $R_\alpha(T) = R(T) + \alpha|T|$
  - $R(T)$: Misclassification error on validation set

Cost Complexity Pruning Algorithm

Systematic approach to find optimal tree size:

- **Cost function**: $R_\alpha(T) = R(T) + \alpha|T|$
  - $R(T)$: Misclassification error on validation set
  - $|T|$: Number of terminal nodes (tree size)

Cost Complexity Pruning Algorithm

Systematic approach to find optimal tree size:

- **Cost function**: $R_\alpha(T) = R(T) + \alpha|T|$
  - $R(T)$: Misclassification error on validation set
  - $|T|$: Number of terminal nodes (tree size)
  - α: Complexity parameter (penalty for larger trees)

Cost Complexity Pruning Algorithm

Systematic approach to find optimal tree size:

- **Cost function**: $R_\alpha(T) = R(T) + \alpha|T|$
  - $R(T)$: Misclassification error on validation set
  - $|T|$: Number of terminal nodes (tree size)
  - α: Complexity parameter (penalty for larger trees)
- **Process**:

Cost Complexity Pruning Algorithm

Systematic approach to find optimal tree size:

- **Cost function**: $R_\alpha(T) = R(T) + \alpha|T|$
  - $R(T)$: Misclassification error on validation set
  - $|T|$: Number of terminal nodes (tree size)
  - α: Complexity parameter (penalty for larger trees)
- **Process**:
  1. Start with full tree ($\alpha = 0$)

Cost Complexity Pruning Algorithm

Systematic approach to find optimal tree size:

- **Cost function**: $R_\alpha(T) = R(T) + \alpha|T|$
  - $R(T)$: Misclassification error on validation set
  - $|T|$: Number of terminal nodes (tree size)
  - α: Complexity parameter (penalty for larger trees)
- **Process**:
  1. Start with full tree ($\alpha = 0$)
  2. Gradually increase α

Cost Complexity Pruning Algorithm

Systematic approach to find optimal tree size:

- **Cost function**: $R_\alpha(T) = R(T) + \alpha|T|$
  - $R(T)$: Misclassification error on validation set
  - $|T|$: Number of terminal nodes (tree size)
  - α: Complexity parameter (penalty for larger trees)
- **Process**:
  1. Start with full tree ($\alpha = 0$)
  2. Gradually increase α
  3. At each α, prune branches that increase cost

Cost Complexity Pruning Algorithm

Systematic approach to find optimal tree size:

- **Cost function**: $R_\alpha(T) = R(T) + \alpha|T|$
  - $R(T)$: Misclassification error on validation set
  - $|T|$: Number of terminal nodes (tree size)
  - α: Complexity parameter (penalty for larger trees)
- **Process**:
  1. Start with full tree ($\alpha = 0$)
  2. Gradually increase α
  3. At each α, prune branches that increase cost
  4. Select α with best cross-validation performance

Bias-Variance Trade-off in Trees

- **Unpruned trees**:

Bias-Variance Trade-off in Trees

- **Unpruned trees**:
  - Low bias (can fit complex patterns)

Bias-Variance Trade-off in Trees

- **Unpruned trees**:
  - Low bias (can fit complex patterns)

Bias-Variance Trade-off in Trees

- **Unpruned trees**:
  - Low bias (can fit complex patterns)
  - High variance (sensitive to training data changes)

Bias-Variance Trade-off in Trees

- **Unpruned trees**:
  - Low bias (can fit complex patterns)
  - High variance (sensitive to training data changes)
  - Prone to overfitting

Bias-Variance Trade-off in Trees

- **Unpruned trees**:
  - Low bias (can fit complex patterns)
  - High variance (sensitive to training data changes)
  - Prone to overfitting
- **Heavily pruned trees**:

Bias-Variance Trade-off in Trees

- **Unpruned trees**:
  - Low bias (can fit complex patterns)
  - High variance (sensitive to training data changes)
  - Prone to overfitting
- **Heavily pruned trees**:
  - High bias (may miss important patterns)

Bias-Variance Trade-off in Trees

- **Unpruned trees**:
  - Low bias (can fit complex patterns)
  - High variance (sensitive to training data changes)
  - Prone to overfitting
- **Heavily pruned trees**:
  - High bias (may miss important patterns)
  - Low variance (more stable predictions)

Bias-Variance Trade-off in Trees

- **Unpruned trees**:
  - Low bias (can fit complex patterns)
  - High variance (sensitive to training data changes)
  - Prone to overfitting
- **Heavily pruned trees**:
  - High bias (may miss important patterns)
  - Low variance (more stable predictions)

Bias-Variance Trade-off in Trees

- **Unpruned trees**:
  - Low bias (can fit complex patterns)
  - High variance (sensitive to training data changes)
  - Prone to overfitting
- **Heavily pruned trees**:
  - High bias (may miss important patterns)
  - Low variance (more stable predictions)
  - Risk of underfitting

Bias-Variance Trade-off in Trees

- **Unpruned trees**:
  - Low bias (can fit complex patterns)
  - High variance (sensitive to training data changes)
  - Prone to overfitting
- **Heavily pruned trees**:
  - High bias (may miss important patterns)
  - Low variance (more stable predictions)
  - Risk of underfitting
- **Optimal pruning**: Balances bias and variance

Bias-Variance Trade-off in Trees

- **Unpruned trees**:
  - Low bias (can fit complex patterns)
  - High variance (sensitive to training data changes)
  - Prone to overfitting
- **Heavily pruned trees**:
  - High bias (may miss important patterns)
  - Low variance (more stable predictions)
  - Risk of underfitting
- **Optimal pruning**: Balances bias and variance
- **Cross-validation**: Essential for finding this balance

Practical Pruning Guidelines

- **Start simple**: Begin with restrictive pre-pruning parameters

Practical Pruning Guidelines

- **Start simple**: Begin with restrictive pre-pruning parameters

Practical Pruning Guidelines

- **Start simple**: Begin with restrictive pre-pruning parameters
- **Cross-validation**: Always use CV to select pruning parameters

Practical Pruning Guidelines

- **Start simple**: Begin with restrictive pre-pruning parameters
- **Cross-validation**: Always use CV to select pruning parameters

Practical Pruning Guidelines

- **Start simple**: Begin with restrictive pre-pruning parameters
- **Cross-validation**: Always use CV to select pruning parameters
- **Validation curves**: Plot training/validation error vs. tree complexity

Practical Pruning Guidelines

- **Start simple**: Begin with restrictive pre-pruning parameters
- **Cross-validation**: Always use CV to select pruning parameters
- **Validation curves**: Plot training/validation error vs. tree complexity

Practical Pruning Guidelines

- **Start simple**: Begin with restrictive pre-pruning parameters
- **Cross-validation**: Always use CV to select pruning parameters
- **Validation curves**: Plot training/validation error vs. tree complexity
- **Common parameters** (sklearn):

Practical Pruning Guidelines

- **Start simple**: Begin with restrictive pre-pruning parameters
- **Cross-validation**: Always use CV to select pruning parameters
- **Validation curves**: Plot training/validation error vs. tree complexity
- **Common parameters** (sklearn):
  - `max_depth`: Start with 3-10

Practical Pruning Guidelines

- **Start simple**: Begin with restrictive pre-pruning parameters
- **Cross-validation**: Always use CV to select pruning parameters
- **Validation curves**: Plot training/validation error vs. tree complexity
- **Common parameters** (sklearn):
  - `max_depth`: Start with 3-10

Practical Pruning Guidelines

- **Start simple**: Begin with restrictive pre-pruning parameters
- **Cross-validation**: Always use CV to select pruning parameters
- **Validation curves**: Plot training/validation error vs. tree complexity
- **Common parameters** (sklearn):
    - `max_depth`: Start with 3-10
    - `min_samples_split`: Try 10-100

Practical Pruning Guidelines

- **Start simple**: Begin with restrictive pre-pruning parameters
- **Cross-validation**: Always use CV to select pruning parameters
- **Validation curves**: Plot training/validation error vs. tree complexity
- **Common parameters** (sklearn):
  - `max_depth`: Start with 3-10
  - `min_samples_split`: Try 10-100
  - `min_samples_leaf`: Try 5-50

Practical Pruning Guidelines

- **Start simple**: Begin with restrictive pre-pruning parameters
- **Cross-validation**: Always use CV to select pruning parameters
- **Validation curves**: Plot training/validation error vs. tree complexity
- **Common parameters** (sklearn):
  - `max_depth`: Start with 3-10
  - `min_samples_split`: Try 10-100
  - `min_samples_leaf`: Try 5-50
  - `ccp_alpha`: Use for cost complexity pruning

Practical Pruning Guidelines

- **Start simple**: Begin with restrictive pre-pruning parameters
- **Cross-validation**: Always use CV to select pruning parameters
- **Validation curves**: Plot training/validation error vs. tree complexity
- **Common parameters** (sklearn):
  - `max_depth`: Start with 3-10
  - `min_samples_split`: Try 10-100
  - `min_samples_leaf`: Try 5-50
  - `ccp_alpha`: Use for cost complexity pruning
- **Domain knowledge**: Consider interpretability requirements

Summary

- Interpretability an important goal

Summary

- Interpretability an important goal
- Decision trees: well known interpretable models

Summary

- Interpretability an important goal
- Decision trees: well known interpretable models

Summary

- Interpretability an important goal
- Decision trees: well known interpretable models
- Learning optimal tree is hard

Summary

- Interpretability an important goal
- Decision trees: well known interpretable models
- Learning optimal tree is hard
- Greedy approach:

Summary

- Interpretability an important goal
- Decision trees: well known interpretable models
- Learning optimal tree is hard
- Greedy approach:
- Recursively split to maximize “performance gain”

Summary

- Interpretability an important goal
- Decision trees: well known interpretable models
- Learning optimal tree is hard
- Greedy approach:
- Recursively split to maximize "performance gain"

Summary

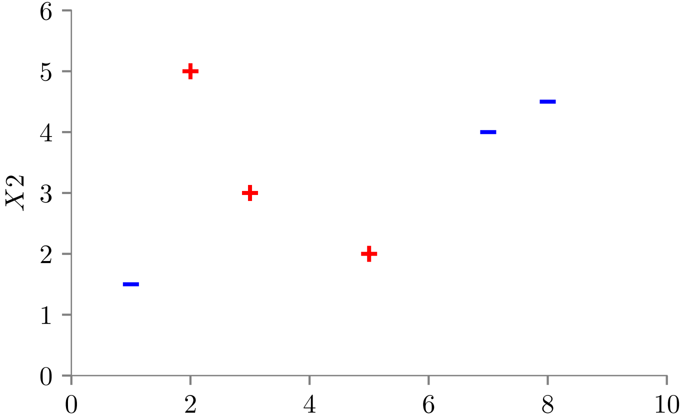
- Interpretability an important goal
- Decision trees: well known interpretable models
- Learning optimal tree is hard
- Greedy approach:
- Recursively split to maximize “performance gain”
- Issues:

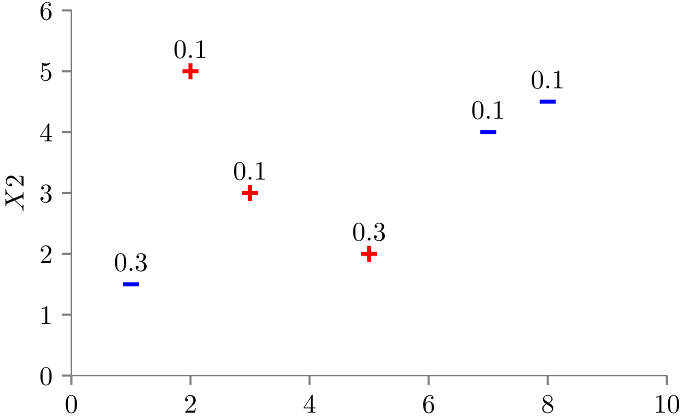
Summary

- Interpretability an important goal
- Decision trees: well known interpretable models
- Learning optimal tree is hard
- Greedy approach:
- Recursively split to maximize "performance gain"
- Issues:
    - Can overfit easily!

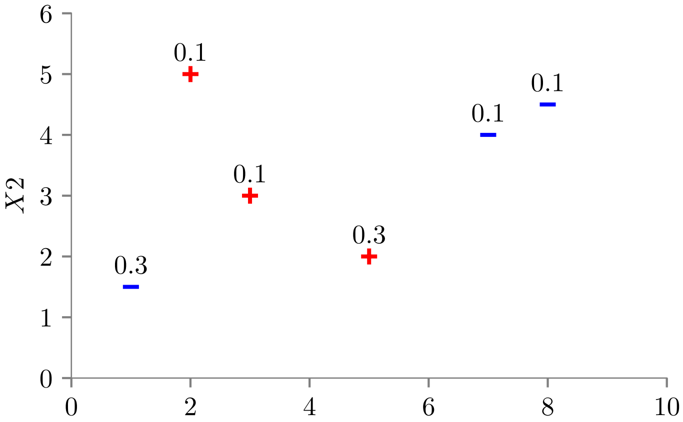
Summary

- Interpretability an important goal
- Decision trees: well known interpretable models
- Learning optimal tree is hard
- Greedy approach:
- Recursively split to maximize “performance gain”
- Issues:
    - Can overfit easily!
    - Empirically not as powerful as other methods


X2
6
5
4
3
2
1
0
0
2
4
6
8
10

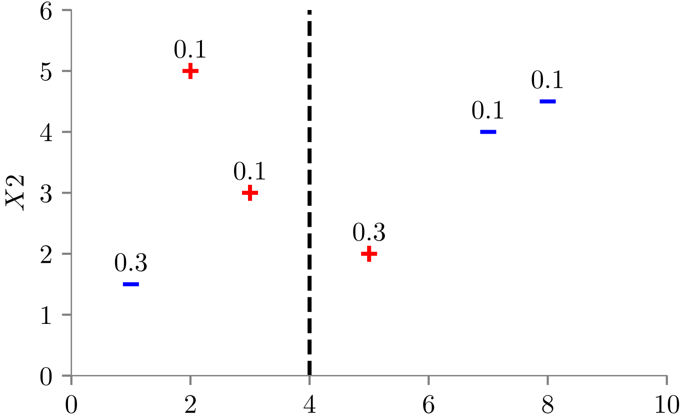


0.1
0.1
0.1
0.3
0.1
0.3
X2
0
1
2
3
4
5
6
0
2
4
6
8
10

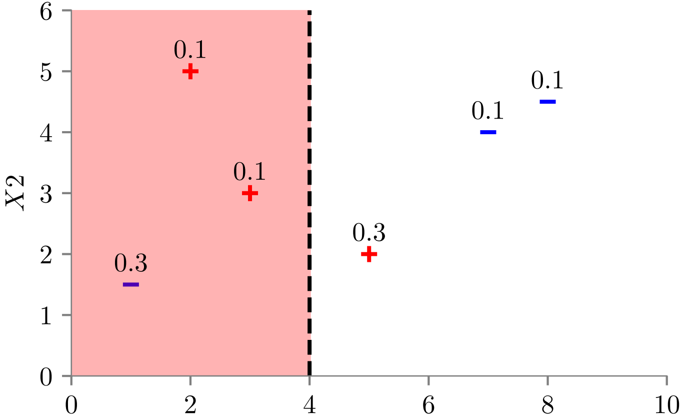


$$\text{Entropy} = -P(+) \log_2 P(+) - P(-) \log_2 P(-)$$

$$P(+) = \frac{0.1 + 0.1 + 0.3}{1} = 0.5, \quad P(-) = \frac{0.3 + 0.1 + 0.1}{1} = 0.5$$



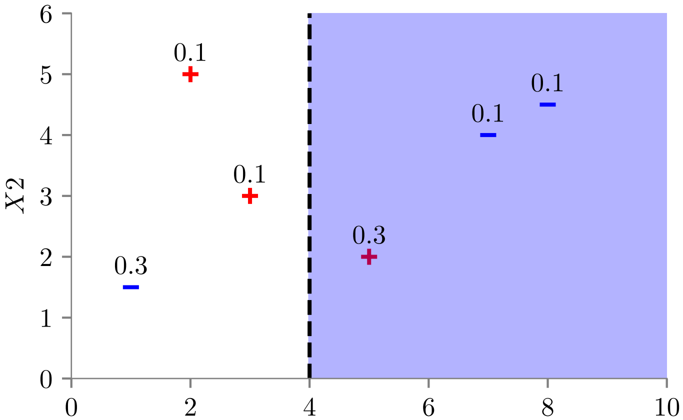
Candidate Line: $X1 = 4(X1^*)$



Entropy of $X1 \leq X1^* = E_{S(X1<X1^*)}$

$$P(+) = \frac{0.1 + 0.1}{0.1 + 0.1 + 0.3} = \frac{2}{5}$$

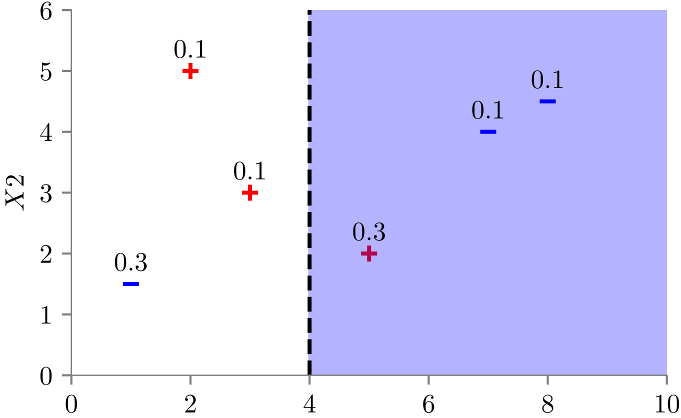
$$P(-) = \frac{3}{5}$$



Entropy of $X_1 > X_1^* = E_{S(X_1 > X_1^*)}$

$$P(+) = \frac{3}{5}$$

$$P(-) = \frac{2}{5}$$



$$\mathsf{IG}(X_1 = X_1^*) = E_S - \frac{0.5}{1} \cdot E_{S(X_1 < X_1^*)} - \frac{0.5}{1} \cdot E_{S(X_1 > X_1^*)}$$